



Europäisches Patentamt
European Patent Office
Office européen des brevets



(11) Publication number:

0 562 215 A1

(12)

EUROPEAN PATENT APPLICATION

(21) Application number: 92810228.4

(51) Int. Cl. 5: H04L 29/06

(22) Date of filing: 27.03.92

(43) Date of publication of application:
29.09.93 Bulletin 93/39

(64) Designated Contracting States:
DE FR GB

(71) Applicant: International Business Machines
Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Mueller, Lothar, Dr.
Rütistrasse 36

CH-8134 Adliswil(CH)
Inventor: Oechsle, Rainer, Dr.

Rütibohistrasse 19

CH-8135 Langnau(CH)
Inventor: Truong, Hong Linh, Dr.
Obstgartenweg 13
CH-8136 Gattikon(CH)

(74) Representative: Barth, Carl Otto et al
IBM Corporation Säumerstrasse 4
CH-8803 Rüschlikon (CH)

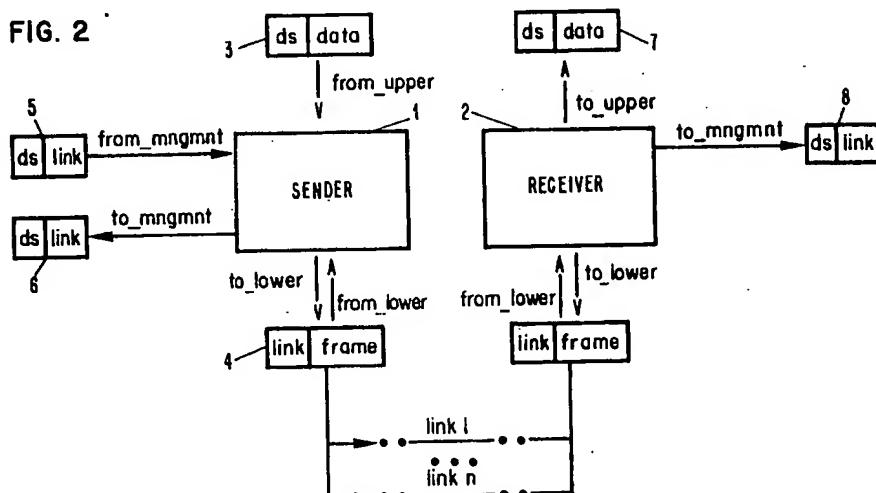
(54) Order preserving transmission of data streams.

(57) Data transmission in a network over two or more parallel links (link 1 . . . link n) between two network nodes (1 and 2), a so-called transmission group, usually requires (if the transmission protocol has no other way to solve the order issue) to preserve the order of the data frames within a data stream if switching from one link to the other occurs within a data stream, or, in other words, if a data stream is switched from one link to another.

Two novel approaches are disclosed.

- The first method achieves the order preservation by numbered switch frames (4) that have to be exchanged between the nodes before the link switching is actually performed.
- In the second method, the switching is only prepared by exchanging special frames which indicate "clean" links. Actual switching takes place by just using an already prepared, i.e. "clean" link.

FIG. 2



EP 0 562 215 A1

Field of the Invention

In general, this invention relates to data transmission in a network and concerns particularly the transmission of data between two nodes within such a network. In such networks, two nodes are often connected by two or more links, be it for the reason of security, of availability or of reliability. If, in such a network, switching from any one link to another (between the same two nodes) occurs within one and the same data stream, in other words, if a data stream is switched from one link to another, it is in many applications imperative that the order of the data frames within said particular data stream is preserved in some way. If this is not achieved, the data will not be received correctly: consequently, a time consuming retransmission of the whole data stream may be necessary or, if the disorder is not detected, data may be corrupted. In other words, the requirement of a single link appearance of a transmission group, i.e. a multi-link connection between nodes, enforces some procedure for preserving the order of the frames.

Background

The problem of order preservation in data streams when switching from one link to another in transmission groups has been recognized before. IBM's System Network Architecture (SNA) is an example. Tutorials on the solution used in SNA are (1) J. Martin: SNA, IBM's Networking Solution, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1987, and (2) A. Meijer: System Network Architecture, A Tutorial, Pittman Publishing, London, UK, 1987. The SNA method is to attach a sequence number to each data frame and reorder the frames at the receiving node according to this number.

Though it is a rather safe method, this solution, however, has disadvantages. The numbering of each and every data frame occupies space and time that is lost for data, i.e. payload transmission. Further, both the sending and the receiving node have to keep track of the numbers given, permanently frame by frame, in other words, there is a substantial administration effort attributable solely to the order preservation requirement.

Further, in order to rearrange potentially unordered frames in the receiver, buffering of frames must be provided which occupies storage resources and needs additional administrative overhead. A particular problem is the wrap-around necessary when the frame numbers reach the upper limit, i.e. when the maximum capacity of the order number field or section in each frame is reached.

More details on the SNA concept are described under the heading Introduction below.

Objects of the Invention

One object of the invention is to overcome the above mentioned deficiencies of the method given in the prior art.

Another object is to devise a method that, given certain conditions that the data link fulfils, achieves order preservation of the transmitted data frames with a minimum of "administrative" data transmission between the two nodes concerned.

A further object is to enable a distribution of the data to be transmitted onto selected links according to practically any governing rule. E.g. according to speed requirements, if links with different transmission speeds are used; according to security requirements in networks where the security within a transmission group varies; or according to economic patterns in a network with differently priced transmission links; or any combination of the above and other rules.

Assuming that data streams can be identified, that it is sufficient to guarantee order preservation only for each data stream and that frame loss is tolerable, the new solution disclosed here for data streams groups is a very adaptive and well controllable way for sending data over transmission groups.

Summary of the Invention

In brief, the invention concerns a new method for link switching when transmitting a data stream between two nodes of a network having multiple parallel links. During the transmission, data streams are assigned to links; the sender transmits data only over a so-called active link and the receiver accepts data only from an active link.

To obtain the desired flexibility, the method for order-preserving shifting of the data stream from one link to another link the link-switching - includes a sequence of basic steps, namely

- generating a special request in the sending node (sender),
- attaching a sequence number to said request,

- transmitting said request with its number to the receiving node (receiver) over at least one currently inactive link,
- determining whether a received request carries a higher sequence number than a request received before (and is thus acceptable),
- 5 - if so, sending to the sender, over said currently inactive link, a special response with the sequence number of said received request, and
- determining, in the sender, when receiving a response, whether its sequence number matches the sequence number sent with the last request and is thus acceptable,
- 10 - depending on the result of at least one determining step, activating a so far inactive link to transmit the data stream.

The subclaims define two possible approaches, both assigning each data stream to a particular link. The first method achieves this by numbered switch frames that first have to be exchanged between the nodes before the link switching actually can take place. In the second method, the switching is only prepared by exchanging special frames. The actual switching takes place by just using an already prepared link. Both solutions are efficient, flexible, and order preserving, thus achieving the objects of the invention.

Introduction

Connecting two adjacent network nodes by more than a single link is an appropriate way of increasing the communication capacity and reliability between the two nodes. Such a set of parallel links is usually called a transmission group. In the prior art, in SNA, for example, the bottom sublayer of the Path Control layer (corresponding to OSI layer 3, the network layer) which is called the Transmission Group sublayer, combines one or more parallel SDLC (Synchronous Data Link Control) links and makes them appear to higher layers as a single link. Consequently the FIFO nature of a single link has to be guaranteed for the transmission group as a whole by the Transmission Group sublayer.

In the SNA approach, this is achieved as follows. A sequence number is attached to each frame at the sender side. At the receiver side, the frames are ordered according to the sequence numbers. If a frame is missing, it is retransmitted over another link until it is successfully transmitted or the maximum retry count is reached. Because of the transmission of the same frame on different links in case of transmission errors, there is the problem that no easy wrap-around of the sequence numbers as in SDLC is possible (sequence numbers are in the range from 0 to 4095). SNA solves this problem by quiescing all links of the transmission group when the maximum number is reached. The receiver first sends a sequence number wrap acknowledgement. The sender then sweeps all links, i.e. makes sure that there are no old frames in the links anymore, sends a frame with sequence number 0 and sweeps the links again. More detailed information on this topic can be found in the textbooks on SNA, cited above, or in other SNA literature.

The overhead of each frame carrying an additional sequence number, the procedures of frame resequencing, retransmission on different links and of the sequence number wrap-around are considerable.

This invention discloses two more efficient solutions for the transmission of data over transmission groups, provided that the assumptions described in the following section hold.

40 Assumptions

The frames to be transmitted over a transmission group can be identified as belonging to different data streams and order preservation has to be guaranteed only for each individual data stream, but not globally for all frames. Such a data stream may be identified by some kind of a connection identifier in the frame or by a pair of end points, as for example the pair destination MAC (Medium Access Control) address and destination LLC SAP (Logic Link Control, Service Access Point), source MAC address and source LLC SAP in case of two LANs being remotely bridged over a transmission group.

As will be seen below, for each data stream a certain amount of state information has to be remembered. So, if the granularity of different recognizable data streams is too fine, the amount of state information may become very large. In this case, the granularity can be made coarser by grouping several data streams together, e.g. by taking into account only the first n most significant bits of a connection identifier or, in case of remote bridging, by identifying a data stream only through the pair (destination MAC address, source MAC address) disregarding the LLC SAPs.

55 A possible loss of frames is tolerable. Thus, no retransmission scheme is implemented. The underlying assumption is that this is done on an end-to-end basis, e.g. by the LLC in case of remote bridging. Note that a protocol that tolerates frame losses is not necessarily also able to deal with frame reordering. LLC may be taken as an example.

A data stream is completely independent from its corresponding reverse data stream. In the following payload data transmission is thus considered only for one direction, from a sender to a receiver node.

General Concept of the Invention

Because of the first assumption mentioned in the previous section, namely that different data streams can be identified and that it is sufficient to preserve the frame order only for each data stream, a simple exploitation of a transmission group is just to distribute the data streams onto the links of the group, typically sending more than one data stream over the same link, but never sending a single data stream over several links (because of order preservation difficulties). What is needed is a simple table that assigns a link to each data stream. An obvious disadvantage of this approach is the loss of flexibility as compared to the method described for SNA. E.g., if one link is heavily loaded because of a tremendous increase of traffic for the data streams assigned to this link while the utilization of other links is rather low, a need arises to balance the traffic load within the transmission group. It is thus required to be able to dynamically shift a data stream from one link to another one without corrupting the order. This action will be called link switching. Here, two methods are described how to achieve such a link switching.

In both methods, the sender and the receiver store for each data stream the currently used link called the active link, and a sequence number counting the number of switching requests for this data stream. The sender transmits the payload data for a given data stream only on the currently active link and the receiver accepts frames for a given data stream only from the currently active link.

In the first solution, explicit switch request frames are sent from the sender to the receiver in order to inform the receiver that the active link shall be changed. Switch request frames are always sent over that link to which the data stream shall be switched, i.e. they are sent over the 'new' link. On receiving such a switch request frame, the receiver changes the active link to that link on which the request frame arrived for the data stream as indicated in the request. In addition, the receiver replies to such switch requests by sending switch responses back also over the 'new' link. The sender then also changes his active link and thereafter starts to forward frames for the corresponding data stream on the new link. Because such switch request and response frames may be lost or heavily delayed, it may be decided to resend them. The loss or delay of switch frames may be taken as an indication that switching to this particular link is not a good idea. So another link may be selected to switch to. This means that switch request frames are resent, possibly over different links. In order to decide at the receiver side which request frame is the most recent one, a sequence number is attached to each switch frame (as it is done in SNA for each data frame). The receiver only switches the link and sends replies after having received a switch request frame with a sequence number higher than the sequence numbers of all previously received frames. This sequence number is also added to each switch response frame.

Similarly, the sender only switches to the new link if a switch response is received with the same sequence number as the last sent switch request. Note that the sequence numbers are handled on a per data stream basis. Note also that frames sent on the old link while the receiver already has switched to the new link, are simply lost. The assumption is, however, that such a loss is tolerable. The typical way of using this method, however, is to stop transmitting until the switch response has arrived.

Thus, in the first solution, a switch request and response first have to be exchanged. The second solution presented just avoids this delay. Instead of exchanging switch request and response frames, the candidate links are swept by exchanging sweep request and response frames. These sweep frames, however, do not trigger the change of the active link for a given data stream, but the reception of such a sweep request and response frame is only taken as an indication by the receiver and the sender, respectively, that the given link is clean and a switch to this link is now possible. The actual switching takes place by simply using from now on a clean link at the sender side. The receiver switches to the new link upon receiving the first frame over a new, clean link. After having switched a link, all links are considered to be dirty again. This is done in the same way at the sender and at the receiver side. As in the previous proposal, sweep requests and responses carry sequence numbers to distinguish old and new sweep frames. The sender's sequence number is increased for each switching action done at the sender side, but not for each sweeping frame sent! The receiver accepts a sweeping frame arriving over a yet dirty link and carrying the current sequence number, i.e. the link is added to the set of clean links at the receiver side. Similarly, the sender adds a link to the set of clean links if a sweep response arrives carrying the current sequence number. This allows that more than one link can be swept and considered clean. There are thus several links ready to potentially take over the data stream which is very important in a critical situation. Like the sender, the receiver increases its sequence number with each switching action done. If the receiver gets a sweep request frame with a sequence number larger than the current one, it notices that one switch

was missed.

Consequently, this case is handled like a switch, i.e. all links are considered dirty except the one for which the sweep arrived. Note that the active link does not always have to be clean. The state of a link determines whether a data stream can be switched to that link or not. This state, however, has no meaning for the active link. So, e.g. in the case mentioned above where the receiver notices a missing switch and considers all links to be dirty except the one for which the sweep arrived, the active link stays the same and the receiver still accepts data frames on this active link until an actual switch takes place, i.e. until a data frame arrives on a different, clean link.

Obviously, in the second solution, a switching can be done immediately if there are clean links available. This advantage, however, has to be paid by storing for each data stream which link is dirty and which one is clean. So there is a clear trade-off between the two solutions. Which method to use can only be decided if the requirements for a special application are known.

It shall be pointed out that there is no method for the wrap-around of the sequence numbers. It is not needed because

15 - the sequence number only counts the switchings per data stream and not globally all individual frames, thus its growing is much smaller, and - because a sequence number is transmitted only in switch or sweep frames,

- it is thus affordable to use as many bits as needed.

16 In the following two sections, a detailed description of the methods outlined above is presented with reference to the drawings.

The Drawings

25 *Fig. 1* shows a typical frame with its various sections used in solution I of the invention;

Fig. 2 shows sender and receiver and their connection according to solution I of the present invention;

Fig. 3 shows a typical frame with its various sections used in solutions II of the present invention;

Fig. 4 shows sender and receiver and their connection according to solution II of the present invention.

30 **Detailed Description of Embodiment I**

The complete method is described by two processes, a sender and a receiver process. They interact with their local system environment by sending and receiving messages 13, 15, 16, etc. to and from operating systems channels as depicted in Fig. 2. (For the terminology: "frames" are exchanged between the two nodes; "messages" are exchanges within a node to control the processes.)

35 Each frame (Fig. 1) exchanged between sender 1 and receiver 2 over the links contains at least three components, preferably in three fields of the frame. First, a type-of-frame indicator (type) that determines which type of frame it is. Three types exist: DATA frames, containing payload data; SWITCH REQUEST frames, identifying a request from a sender, explained in detail below; and SWITCH RESPONSE frames, identifying a response from a receiver on such a sender request, also explained below in detail. Second, a data stream identifier (ds) that identifies the data stream to which the frame belongs. Third, an information part (info) which either contains the payload data if the frame is of the DATA type, or a sequence number if the frame is of the SWITCH type.

40 Sender 1 (Fig. 2) receives messages 3 from the upper layer through the upper channel. Messages 3 contain a data stream identifier (ds) and the payload (data) to be transmitted. In addition, the sender process sends and receives messages 4 to and from the lower layer through the channels to lower and from lower, respectively. These messages contain a link section (link) and a frame (frame) according to Fig. 1. The meaning of such a message is that the frame is to be sent or was received over the given link. The frame itself carries the type field, indicating whether it is a (pure) data frame or a switch request/response frame, the data stream identification and an information field which is interpreted as payload for data frames and as a sequence number for switch frames.

45 Also, a management process may send switch request messages 5 to the sender process through the channel from mngmnt. These messages 5 simply contain the data stream identifier (ds) and the new link (link). Whenever a switch has been done successfully, this is reported to the management by messages 6 through the channel to mngmnt.

50 The interaction of receiver 2 with its environment is very similar to the sender process with the exception that receiver 2 sends messages 7 to the upper layer, via the channel to upper, instead of

receiving from it. Each message 7 is an equivalent to message 3 that the sender received from the upper layer. Also, receiver 2 reports switching actions by messages 8 to the management process (via the channel to management) but does not accept any commands from it. These messages 8 are equivalent to the messages 6 on the sender side.

5 In the following, the relevant control information held at the sender and at the receiver side together with its initialization is shown in C syntax:

```

10     typedef struct {LINK active_link; NUMBER seqno} cntrl;
         cntrl sendcntrl [MAX_DS],receivecntrl [MAX_DS];

15     init()
     {
         for{i = 0; i < MAX_DS; i++)
         {
             sendcntrl [i].active_link = initial_link;
             sendcntrl [i].seqno = 0;
             receivecntrl [i].active_link = initial_link;
             receivecntrl [i].seqno = 0;
         }
     }

```

The sender and receiver processes are also written in C with the following extensions.

- **Send()** and **receive()** calls are used for sending and receiving to and from the operating systems channels. In order to wait for the arrival of a message on more than one channel, the notation $ch = receive(ch1 \text{ or } ch2 \text{ or } \dots \text{ chn}, msg);$ is used where ch gives the channel on which the message finally was received.
- Depending from which channel a message was received, the components may be accessed through the component names as introduced in Fig 1. E.g., if the message msg was received from channel $from_upper$, its components are $msg.ds$ and $msg.data$, whereas the components of a message msg received from $from_lower$ are $msg.link$ and $msg.frame$.
- A message to be sent is simply indicated by giving its components enclosed in $<\dots>$.

Note also that there is no timeout and retry built into the code of the sender process in order to protect against lost switch request or response frames. This timeout supervision and triggering of retries has to be all done by the management process. This simplifies the sender code in a significant way. Here is the sender process first.

```

40    process sender()
41    {
42        forever
43        {
44            received = receive(from_mngmnt or from_upper or ...
45            ... from_lower, msg);
46            if(received == from_upper)
47            // just send the data on the active link
48            {
49                ds = msg.ds;
50                info = msg.data;
51                frame = <DATA,ds,info>;
52                link = sendcntrl[ds].active_link;
53                send(to_lower,<link,frame>);
54            }
55            else if(received == from_mngmnt)
56            // initiate outgoing link switching
57            {
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
995
996
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1202
1203
1204
1205
1206
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1302
1303
1304
1305
1306
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2000
2001
2002
2003
2004
2005
2006
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2098
2099
2099
2100
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2198
2199
2199
2200
2200
2201
2202
2203
2204
2205
2206
2207
2208
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2298
2299
2299
2300
2300
2301
2302
2303
2304
2305
2306
2307
2308
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
23
```

```

5           ds = msg.ds;
           sendcntrl[ds].seqno++;
           seqno = sendcntrl[ds].seqno;
           frame = <SWITCH_REQ,ds,seqno>;
           new_link = msg.link;
           send(to_lower,<new_link,frame>);

10          }
10          else if(received == from_lower)
10          // outgoing link switching has been done
10          {
10              new_link = msg.link;
10              frame = msg.frame;
10              ds = frame.ds;
10              if((frame.type == SWITCH_RES)
10                 && (frame.info == receivecntrl[ds].seqno))
10                 // correct frame type and correct sequence number
10                 {
15                 sendcntrl[ds].active_link = new_link;
15                 send(to_mngmnt,<ds,new_link>);
15             }
15         }
15     }
20

```

And here follows the receiver process, again in C syntax.

```

25         process receiver()
25         {
25             forever
25             {
25                 receive(from_lower,msg);
25                 link = msg.link;
25                 frame_in = msg.frame;
25                 ds = frame_in.ds;
30                 if(frame_in.type == DATA)
30                 // data has been received -> pass to user if received on
30                 the active link
30                 {
35                     if(link == receivecntrl[ds].active_link)
35                     send(to_upper,<ds,frame_in.info>);
35                 }
35                 else if(frame_in.type == SWITCH_REQ)
35                 // switch incoming active link if correct sequence number
35                 {
35                     if(frame_in.info > sendcntrl[ds].seqno)
35                     {
40                         receivecntrl[ds].seqno = frame_in.info;
40                         receivecntrl[ds].active_link = link;
40                         frame_out = <SWITCH_RES,ds,frame_in.info>;
40                         send(to_lower,<link,frame_out>);
40                         send(to_mngmnt,<ds,link>);
40                     }
40                 }
45             }
45         }

```

50 This completes the description of embodiment I.

Detailed Description of Embodiment II

In this section, the second solution is presented (Figs. 3 and 4). The main difference with respect to the 55 interaction of the processes with their local system environment is that the messages exchanged with the management process now have an additional component or field (cmd/event) indicating whether it has to be interpreted as a sweep or switch message. In addition, the types of the frames transmitted over the links are now either data frames, or sweep request/sweep response frames instead of switch frames.

Again, the complete method is described by two processes, a sender and a receiver process. They interact with their local system environment by sending and receiving frames shown in Fig. 3 to and from operating systems channels as depicted in Fig. 4.

Each frame, as shown in Fig. 3, contains at least three components, preferably in three fields of the frame. First, there is a type-of-frame indicator (type) that determines which type of frame it is. Three types exist: DATA frames, containing payload data; SWEEP REQUEST frames, identifying a particular request from a sender, explained in detail below; and SWEEP RESPONSE frames, identifying a response from a receiver on such a sender request, also explained below in detail. Second, a data stream identifier (ds) that identifies the data stream to which the frame belongs. Third, an information part (info) which either contains the payload data, if the frame is of the DATA type, or a sequence number, if the frame is of the SWEEP type.

The interaction of the two processes, the sender and the receiver process, with their upper and lower layers is exactly the same as described for the first embodiment in connection with Figs. 1 and 2, with the following important exception. As indicated above, the type field of each frame shown in Fig. 3 can have one of the values DATA, SWEEP_REQ, or SWEEP_RES (instead of DATA, SWITCH_REQ, or SWITCH_RES as in embodiment I). The obvious reason for this is that in the second solution no switch requests and responses are exchanged, but sweep requests and responses. As before, data type frames carry data in the info field, and sweep request and response frames carry a sequence number in the info field.

Thus, an important difference to the previous embodiment I is the interaction of sender 11 and receiver 12 with their management processes. This is due to the fact that management now can give two different commands to the sender process: one (SWEEP) that requests sender 11 to sweep a link for a certain data stream, and another command (SWITCH) which actually switches a data stream to a new link. For that purpose, messages 15 sent from the management process via the from_management channel have an additional command field cmd, namely to distinguish sweep (SWEEP) and switch requests (SWITCH). A sweep request triggers the sender to exchange sweep request and response messages with its receiver process, whereas the switch request results in a purely local action, namely to use the indicated link for future outgoing frames belonging to the indicated data stream (provided the link is clean). Likewise, the messages sent from sender 11 and receiver 12 to their management processes also have an additional event field (EVENT) to indicate if a sweep or a switch has been successfully carried out.

Sender 11 (Fig. 4) receives messages 13 from the upper layer through the from_upper channel. Each message 13 contains a data stream identifier (ds) and the payload (data) to be transmitted. In addition, the sender process sends and receives messages 14 to and from the lower layer through the channels to_lower and from_lower, respectively. These messages contain a link section (link) and a frame (frame) according to Fig. 3. The meaning of such a message is that the frame is to be sent or was received over the given link. The frame itself carries the type field, indicating whether it is a (pure) data frame or a sweep request/response frame, the data stream identification and an information field which is interpreted as payload for data frames and as a sequence number for sweep frames.

As already addressed above, a management process may send sweep request messages 15 to the sender process through the channel from_mngmnt. Other than in the first solution, these messages 15 contain the data stream identifier (ds), the new link (link), and a command (cmd), indicating, whether a switch or sweep is to be made. Whenever a switch or sweep has been done successfully, this is reported to the management by a message 16 through the channel to_mngmnt. This message 16 carries an event field, the contents of which (either SWEEP or SWITCH) indicates to the sender management that either a sweep or a switch was carried out.

The interaction of receiver 12 with its environment is again very similar to the sender process with the exception that receiver 12 sends messages 17 to the upper layer, via the channel to_upper, instead of receiving from it. Each message 17 is an equivalent to message 13 that sender 11 receives from the upper layer. Also, receiver 12 reports switching actions by messages 18 to the management process (via the channel to_management) but does not accept any commands from it. These messages 18 are equivalent to the messages 16 on the sender side and carry the respective information (either SWEEP or SWITCH) in an event field.

The control information stored at each side now contains in addition for each data stream the state of the links (clean or dirty). This information and its initialization is shown below, as usual in C notation. In the following, the relevant control information held at the sender and at the receiver side together with its initialization is shown in C syntax:

```

5      typedef struct {
6          LINK active_link;
7          NUMBER seqno;
8          enum {DIRTY,CLEAN} state[MAX_LINKS];
9      } cntrl;
10
11      cntrl sendcntrl[MAX_DS],receivecntrl[MAX_DS];
12
13      init()
14  {
15          for(i = 0; i < MAX_DS; i++)
16          {
17              sendcntrl[i].active_link = initial_link;
18              sendcntrl[i].seqno = 0;
19              receivecntrl[i].active_link = initial_link;
20              receivecntrl[i].seqno = 0;
21              for(j = 0; j < MAX_LINKS; j++)
22              {
23                  sendcntrl[i].state[j] = CLEAN;
24                  receivecntrl[i].state[j] = CLEAN;
25              }
26          }
27      }

```

Finally the sender and receiver processes are as follows:

```

25      process sender()
26  {
27      forever
28  {
29      received = receive(from_mngmnt or from_upper
30          or from_lower, msg);
31      if(received == from_upper)
32          // just send the data on the active link
33  {
34      ds = msg.ds;
35      info = msg.data;
36      frame = <DATA,ds,info>;
37      link = sendcntrl[ds].active_link;
38      send(to_lower,<link,frame>);
39  }
40  else if(received == from_mngmnt)
41      // switch to clean link or initiate link sweeping
42  {

```

45

50

55

```

if(msg.cmd == SWITCH)
{
    link = msg.link;
    ds = msg.ds;
    if(sendcntrl[ds].state[link] == CLEAN)
    {
        sendcntrl[ds].seqno++;
        sendcntrl[ds].active_link = link;
        for(j = 0; j < MAX_LINKS; j++)
            sendcntrl[ds].state[j] = DIRTY;
        send(to_mngmnt,<SWITCH,ds,link>);
    }
}
else if(msg.cmd == SWEEP)
{
    link = msg.link;
    ds = msg.ds;
    seqno = sendcntrl[ds].seqno;
    frame = <SWEEP_REQ,ds,seqno>;
    send(to_lower,<link,frame>);
}
else if(received == from_lower)
// outgoing link sweeping has been done
{
    link = msg.link;
    frame = msg.frame;
    ds = frame.ds;
    if(frame.type == SWEEP_RES)
        && (frame.info == sendcntrl[ds].seqno))
    // correct frame type and correct sequence number
    {
        sendcntrl[ds].state[link] = CLEAN;
        send(to_mngmnt,<SWEEP,ds,link>);
    }
}
}

process receiver()
{
    forever
    {
        receive(from_lower,msg);
        link = msg.link;
        frame = msg.frame;
        ds = frame.ds;
        if(frame.type == DATA)
        // data has been received -> pass to user if received on
        the active link
        {
            if(link == receivecntrl[ds].active_link)
                send(to_upper,<ds,frame.info>);
            else if(receivecntrl[ds].state[link] == CLEAN)
            {
                receivecntrl[ds].seqno++; receivecntrl[ds].active_link =
                link;
                for(j = 0; j < MAX_LINKS; j++)
                    receivecntrl[ds].state[j] = DIRTY;
                send(to_mngmnt,<SWITCH,ds,link>);
            }
        }
        else if(frame.type == SWEEP_REQ)
        // sweep incoming active link if correct sequence number {
    }
}

```

```

15         if(frame.info >= sendcntr1[ds].seqno)
16         {
17             if(frame.info > sendcntr1[ds].seqno)
18             {
19                 receivecntr1[ds].seqno = frame.info;
20                 for(j = 0; j < MAX_LINKS; j++)
21                     receivecntr1[ds].state[j] = DIRTY;
22             }
23             receivecntr1[ds].state[link] = CLEAN;
24             frame = <SWEEP_RES,ds,frame.info>;
25             send(to_lower,<link,frame>); send(to_mngmnt,<ds,link>);
26         }
27     }
28 }

```

15 To summarize, the disclosed transmission groups are an appropriate way to increase the communication capacity and reliability between two network nodes. The requirement of a single link appearance of such a group enforces some procedure for preserving the order of the frames. The prior art SNA method is to attach a sequence number to each frame and reorder the frames at the other side according to this number.

20 Under the assumption that several data streams can be identified, that it is sufficient to guarantee order preservation only for each data stream and that frame loss is tolerable, the new approaches allow a flexible and efficient transmission of data streams over transmission groups. Both approaches assign each data stream to a link. For flexibility reasons, a possibility is added that allows an order preserving shifting of a data stream from one link to another one which is called link switching. The first method achieves this by numbered switch frames that first have to be exchanged between the nodes before the link switching actually can take place. In the second, faster method, the switching is only prepared by exchanging special frames. The actual switching takes place by just using an already prepared link. So the second method allows immediate switching, provided the preparation has been done, but it requires more state information for each data stream to be stored.

30 So a trade-off exists between both methods and depending on the environment, one or the other may be preferable. Apart from that, both solutions or embodiments presented are

- efficient, because the overhead carried in each frame is zero (the data stream identifier is assumed to be there anyway) and because the additional protocol for link switching is simple and requires only a very limited amount of frames to be exchanged,
- flexible, because a dynamic change of the assignment of data streams to links is possible by link switching, and
- order preserving as has been shown.

35 The invention has been described with reference to two preferred embodiments. It is, however, understood that a person skilled in the art may vary and modify details, by using the teachings of this description, without departing from the spirit and scope of this invention, as defined in the appended claims.

Claims

1. Method for link switching when transmitting a data stream between two nodes (1 and 2, 11 and 12) of a network such that the order within the data stream is preserved, said two nodes being connected by at least two parallel links (link 1 ... link n), of which only one is currently active, the method comprising the steps of
 - a. generating a switch/sweep request in the sending node (sender 1, 11),
 - b. attaching a sequence number to said switch/sweep request,
 - c. transmitting said switch/sweep request with its sequence number from the sender to the receiving node (receiver 2, 12) over at least one currently inactive link,
 - d. determining, in the receiver (2, 12), whether a received switch/sweep request carries a higher sequence number than a switch/sweep request received before and is thus acceptable,
 - e. if so, sending to the sender (1, 11) over said currently inactive link a switch/sweep response with the sequence number of said received switch/sweep request, and
 - f. determining, in the sender (1, 11), when receiving a switch/sweep response, whether its sequence number matches the sequence number sent with the last switch/sweep request and is thus acceptable,

g. depending on the result of at least one determining step, activating a so far inactive link to transmit the data stream.

2. Method according to claim 1, wherein

5 the receiver (2) switches to said currently inactive link upon determining that a received (switch) request is acceptable, and

the sender (1) switches to said currently inactive link upon determining that a received (switch) response is acceptable,

thus effectively switching the data transmission to said new, so far inactive link.

10 3. Method according to claim 2, wherein the sender (1) continues sending data over the currently active link until the sender switches to the currently inactive link.

15 4. Method according to claim 2, wherein the sender (1), upon sending the (switch) request over the currently inactive link, suspends sending data and, upon receiving an acceptable (switch) response, resumes sending data over the so far inactive link.

20 5. Method according to claim 1, wherein

receipt of an acceptable (sweep) request over a currently inactive link by the receiver (12) indicates to the receiver that this particular link is "clean",

receipt of an acceptable (sweep) response over a currently inactive link by the sender (11) indicates to the sender that this particular link is "clean",

the sender (11) effectively switches data stream transmission to any one of the so far inactive links at any time after the sender has identified this particular link as "clean", and,

25 upon receiving data from a "clean", so far inactive link, the receiver (12) effectively switches data stream reception to said link,

upon establishing data stream transmission over said so far inactive link indicating to the sender and to the receiver that all other links are no more "clean".

30 6. Method according to claim 5, wherein

(sweep) requests and responses (Fig.3) carrying the same sequence number are exchanged over a plurality of currently inactive links, and

each acceptable (sweep) request/response pair is taken as indicator that the particular link over which it was exchanged is "clean".

35 7. Method according to claim 6, wherein

transmission of data is switched by the sender randomly to any one of said "clean", so far inactive links.

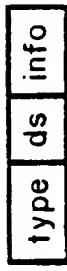
40

45

50

55

FIG. 1



type : DATA, SWITCH_REQ, SWITCH_RES
 info : user data if type = DATA
 sequence no. if type = SWITCH_xxx

FIG. 2

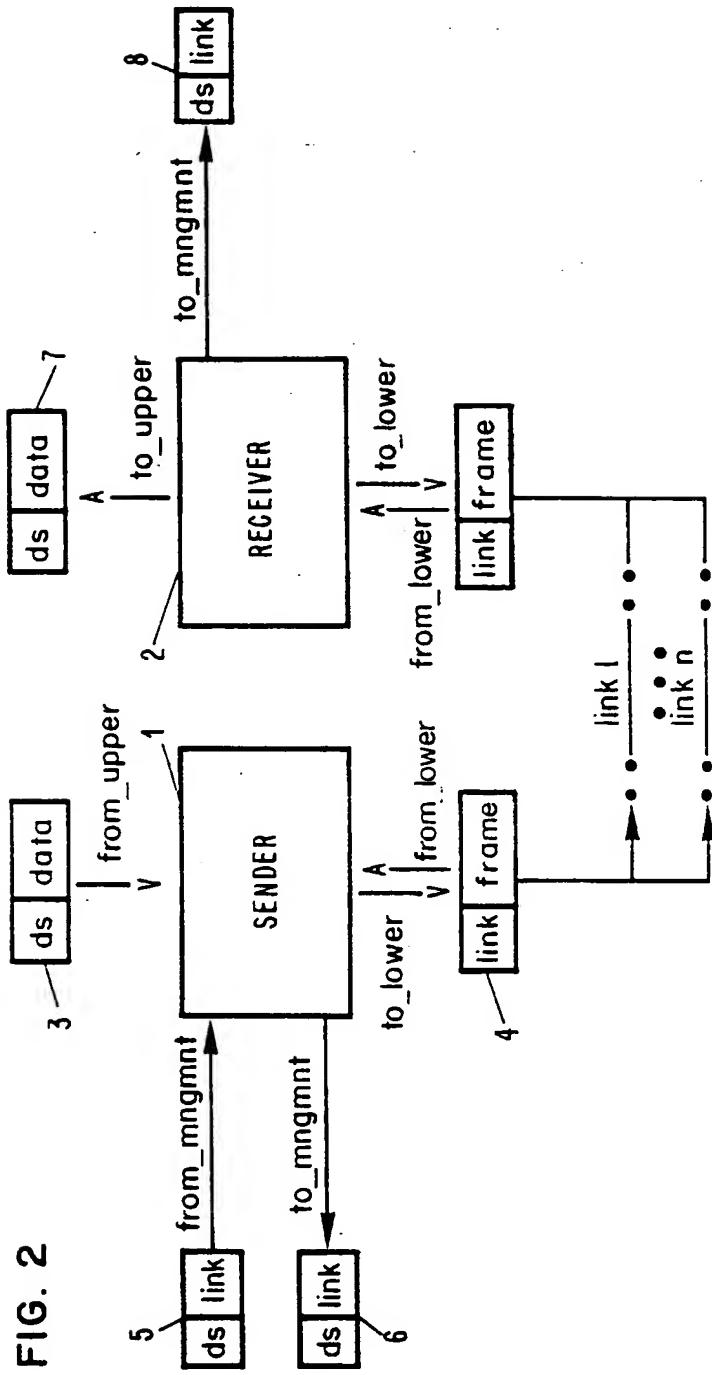
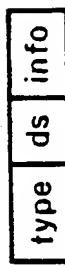
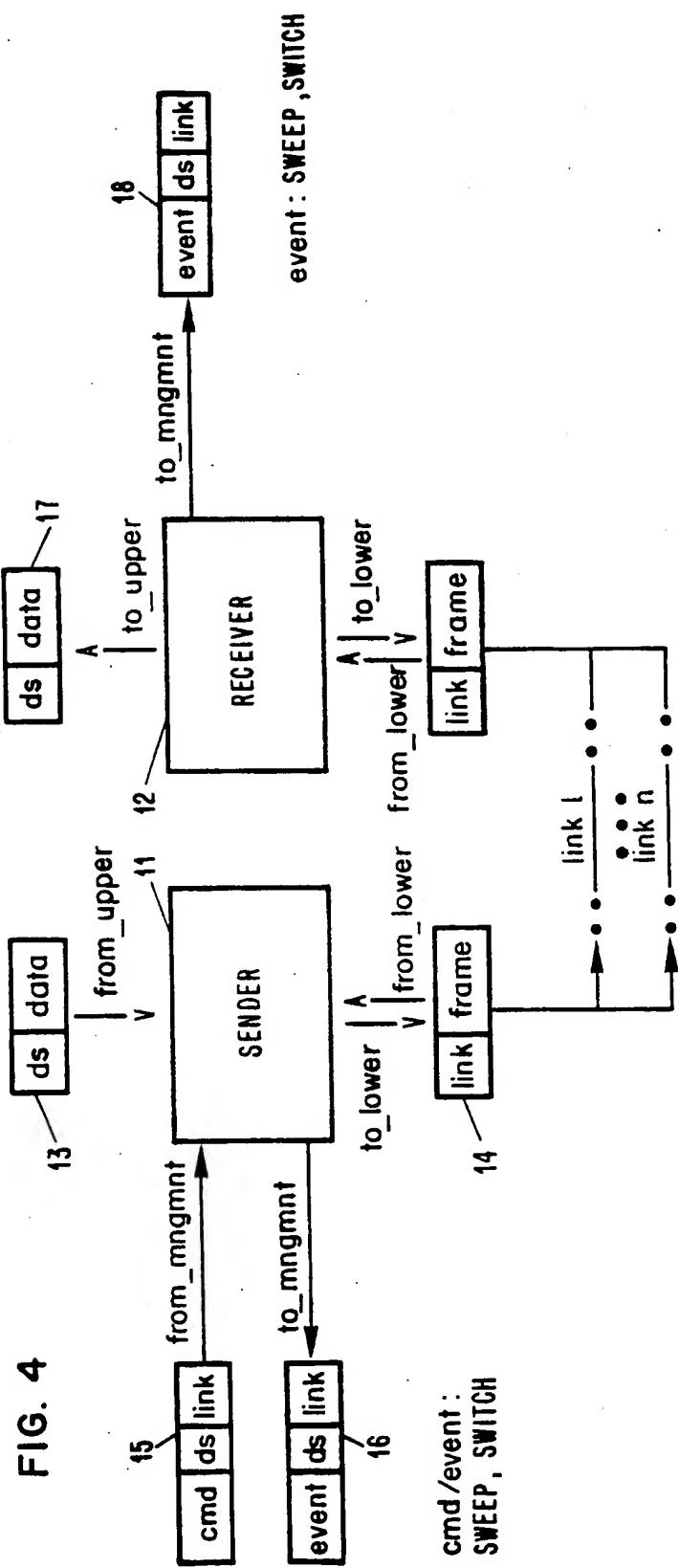


FIG. 3



type : DATA, SWEEP_REQ, SWEEP_RES
 info : user data if type = DATA
 sequence no. if type = SWEEP_XXX

FIG. 4





European Patent
Office

EUROPEAN SEARCH REPORT

Application Number

EP 92 81 0228

DOCUMENTS CONSIDERED TO BE RELEVANT			CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	
A	EP-A-0 224 895 (ATT) * page 3, line 13 - page 7, line 23 * * page 11, line 22 - page 15, line 10 * -----	1-7	H04L29/06
A	WO-A-8 504 300 (BRITISH TELECOM) * page 2, line 23 - page 3, line 3 * * abstract * * figure 1 * -----	1-7	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)
			H04L
<p>The present search report has been drawn up for all claims</p>			
Place of search	Date of completion of the search		Examiner
THE HAGUE	18 NOVEMBER 1992		CANOSA ARESTE C.
CATEGORY OF CITED DOCUMENTS		T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application I : document cited for other reasons & : member of the same patent family, corresponding document	
X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document			